Comparison of High-Performance Packet Processing Frameworks on NUMA

Haipeng Wang and Dazhong He
Center for Data Science,
Beijing Laboratory of Advanced Information Network,
Beijing University of Posts and Telecommunications,
Beijing, China
{neptuntiansea & hdz}@bupt.edu.cn

Huan Wang
HAOHAN Data Technology Co., LTD.
Beijing, China
wanghuan@haohandata.com.cn

Abstract—**Recent studies have been concentrated on comparison of forwarding ability in multiple dimensions among different high-speed packet processing frameworks, such as PF_RING and DPDK, on the whole where transmitting and receiving impact each other, without enough study on frameworks' ability for receiving or transmitting separately. Characteristics of multithreading is another hot spot as multi-core platforms play more important roles. It requires more microscopic insight into comparison of these frameworks' characteristics on multi-core platform, especially on Non-Uniform Memory Access Architecture (NUMA). This paper mainly compared DPDK and PF_RING's processing ability focusing on receiving and multithreading characteristics without transmitting in run-to-completion model and pipeline model, i.e. a kind of load balancing model. Some characteristics about how performance of PF_RING ZC and DPDK in pipeline model specifically on NUMA varies with distribution type of threads among NUMA nodes, were made insight into and concluded. For studying above, a new indicator, ρ, giving the ratio of CPU time consumed on processing per unit packets, was put forward and played an important role in above analysis. By means of ρ, one of this paper's discovery shows that two frameworks' threads' distributions types among NUMA nodes in the pipeline model contribute to different performance of traffic processing.**

key words: DPDK; PF_RING; NUMA; performance; run to completion; pipeline; load balance

I. Introduction

A. Background and Problem Statement

Traditional packet process methods such as NAPI is not adapted to higher-rate network environment. In place, new technologies for higher-rate packet processing have been put forward, such as polling without interrupts, user mode driver, pthread affinity, huge pages, etc. [3].

PF_RING and Data Plane Development Kit (DPDK) are two products used widely adopting these new technologies partly same and partly different. A part of PF_RING's source is open, but the rest such as the most high-rate part, such as PF_RING ZC (zero copy) lib is still closed [8]. Intel DPDK is used more and more widely with favorable comment, providing basic environment, abundant network drivers and function library with source opened for high-rate processing and powerful function [6].

Developers are usually interested in difference of these high-rate frameworks. But recent studies commonly concentrated on the comparison of frameworks' forwarding ability on the whole. E.g. [1] and [10] made comparison of forwarding ability among DPDK, PF_RING ZC, net-map on the whole in multiple dimensions, such as CPU Frequency, processing load, burst size, memory latency, etc. Unfortunately, transmitting and receiving impacts each other for resource contest. At the same time, multithreading performance of these frameworks, such as pipeline, is a hot spot, for multi-core platform play more important roles. Thus, it requires more microscopic insights into comparison of these frameworks' characteristics on multi-core platform, especially on Non-Uniform Memory Access Architecture (NUMA).

B. Major Contributor

This paper didn't study these frameworks' forwarding ability on the whole, but concentrated on receiving, multithreading and multicore application on NUMA, where all test programs were rid of transmitting functions for advoiding impacts to receiving. In these aspects, comparisons between PF_RING and DPDK were conducted and analyzed. At first, this paper compared performance between them in single-threaded model and in multi-threaded pipeline model. Then it compared and analyzed two frameworks' pipeline model specifically on NUMA further. Some analysis and interesting conclusions were made, of which some are inconsistent with relevant studies, e.g. PF_RING ZC is faster than DPDK because of PF_RING ZC's more efficient polling process, in spite of PF_RING ZC's worse inter-thread communication. Some interesting characteristics of these two's pipeline model on NUMA were discovered when threads' distribution among NUMA nodes as a factor varied. At the same time, a new indicator, marked as ρ, giving the ratio of CPU time consumed on processing per unit packets, was put forward to evaluate the processing ability. In equation form, this is

$$\rho = R / V, \tag{1}$$

where ρ is the new indicator described as above, R is the total ratio of CPU time on all packets in a unit time, V is packet processing rate. The unit of ρ *is* the ratio of per Gbps, i.e. %/Gbps,

978-1-4673-9904-3/16/$31.00 ©2016 IEEE

C. *Paper Organization*

The rest of this paper is organized as follows. In Section 2, we firstly described test environment and made essential description of PF_RING and DPDK tested. In Section 3, PF_RING and DPDK's packet processing performance were compared in single thread. In Section 4, the paper tested their multi-thread performance in multi-thread pipeline model, compared with single-thread run-to-completion model. In Section 5, pipeline model on NUMA were compared further, the impacts of threads' distribution among NUMA nodes to performance were studied the time. Finally, summary is drawn in Section 6.

II. EXPERIMENT ENVIRONMENT

This section presents the environment with the software and hardware used. The main environment included a traffic generator and a traffic receiver, of which CPU, i.e. Intel Xeon CPU E5-2603 v2, had no hyper-threading technology but was in NUMA. The traffic generator and the traffic receiver had the same configuration shown as Table I. The connection of the test servers shown in Fig. 1 was direct connection between the servers without the use of hubs or switches to provide measurements unaltered by possible influences of these interconnecting devices.

Intel Xeon CPU E5-2603 v2, which is on NUMA, had no hyper-threading technology. Thus, the total number of logical cores on a physical core is only one. A Server had 2 NUMA nodes with 4 physical/logical cores on per one.

82599ES' feature of Receive Side Scaling (RSS) provides load balancing function for full use of multi-core CPU's parallel processing ability [4], which is helpless to analyze frameworks ability. This paper declined 82599ES' queue to only one for most insight into these frameworks processing performance on per NIC queue. At the same time, the traffic processor's RPS/RFS function was disabled, which provides load balancing in kernel mode. So when traffic receiver was tested for receiving traffic in kernel mode, there was no RPS/RFS, resulting that hardware interrupts and software interrupts run on the same core.

This paper mainly compared two frameworks, namely PF_RING and DPDK described as Table II. All of them were compiled in *-O2* flag. A part of PF_RING's source is open, the rest is not such as PF_RING ZC which is PF_RING's most high-rate part [9].

Packets of different lengths are used for testing (64 B, 128 B, 256 B, 512 B, 768 B, 1024 B, 1280 B and 1500 B). 1518 B Packet is recommended by REF2544, but for the restriction of traffic generator, i.e. zsend based on PF_RING ZC, 1500 is instead. The length does not include 8 bytes Preamble, 4 bytes CRC and 12 bytes IFG.

VTune Amplifier developed by Intel provides a rich set of performance insight into CPU & GPU performance, threading performance & scalability, bandwidth, caching and much more [7]. VTune was deployed to record functions' CPU consumption.

TABLE I. TEST ENVIRONMENT

Item	Configuration
Mainboard	Dell Inc. 068CDY A01 v2.2
Memory	DDR3,1333MHz,16GB
CPU	Intel(R) Xeon(R) CPU E5-2603 v2 @ 1.80GHz
NIC	Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection
OS	CentOS release 6.6 (Final)
Linux kernel	2.6.32-504.12.2.el6.x86_64

TABLE II. VERIONS OF PF_RING AND DPDK

Framework	Version
PF_RING	6.2.0
DPDK	1.8.0

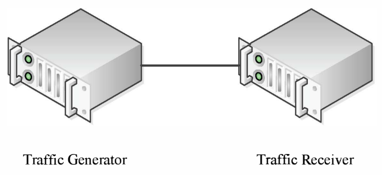


Figure 1. Concection of Servers

III. COMPARISON ON RUN-TO-COMPLETION MODEL

Run-to-completion is a common traffic processing model, which makes programming on multi-core platform more simple and more efficient. This section compared run-to-completion model between PF_RING and DPDK. In this model, packets usually go through receiving, processing, and transmitting.

In these tests, the application as receivers run in single thread, of which affinity set to core 0. And processes were rid of transmitting function reasoned in section 1. All packets were processed as Fig. 2. A software named zsend based on PF_RING ZC was deployed as traffic generator, which generated packets of 64, 128, 256, 512, 768, 1024, 1280 and 1500 referred as section 2. The test result was shown in Fig. 3.

PF_RING ZC's processing rate reached line rate 10Gbps for all packet lengths, except for 64 byte packets only 9.88Gbps was reached. The cause is that the generator only sent a 9.88Gbps rate, 14.03Mpps. So there is a possibility that PF_RING ZC is able to reach 10Gbps for 64B here.

Partly similar to PF_RING ZC, DPDK reached line rate 10Gbps for all packet lengths, except for 64 only 8.18Gbps，11.62Mpps was reached.

With the help of VTune, it was shown that when receiving 64 B packets, DPDK consumed 78.2% of CPU time polling NIC, on which PF_RING ZC 88%. DPDK consumed 19.9% of CPU time on freeing, more exactly recycling, packet buffers back to mempool [6], but PF_RING ZC 0% for this. No recycling operations were found in PF_RING ZC when testing. No smaller details were found for PF_RING ZC is closed.

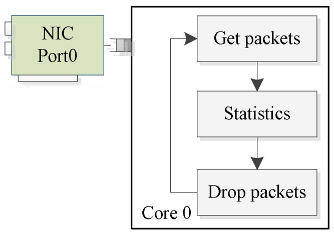


Figure 2. Processing Flow of Packets in Run-to-completion Model

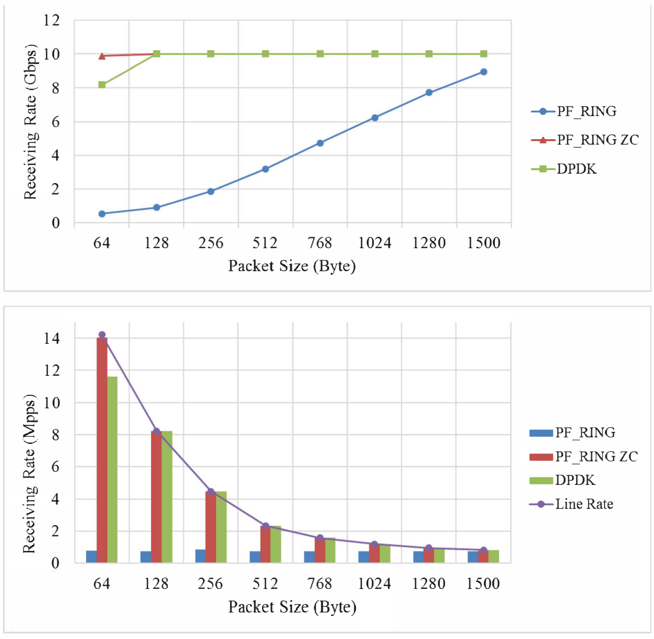


Figure 3. Receiving Rate of Run-to-completion Model

Compared with PF_RING ZC and DPDK, PF_RING (non-ZC) run in a lower rate between 7.3 and 8.3 Mpps for all length packet. It's because PF_RING (non-ZC)'s kernel module received packets in kernel mode so slowly with 100% of CPU time consumed for software interrupt context of one core. It is different with PF_RING ZC and DPDK polling NIC in user mode that time.

IV. Comparison on Pipeline Model

A. Test Description

Pipeline model is another common traffic processing model, separating different parts of a process into different thread modules, which are connected into complex function networks by rings. This section compared PF_RING ZC and DPDK in a pipeline module, as well as a kind of load balancer module.

The pipeline model was configured one RX thread polling the only queue of NIC and two worker threads receiving packets from rings connected to RX thread. Rx thread allocated the mempool for packet buffering on the NUMA node where it is. RX thread got packets' 30th byte data filled by NIC as a RSS hash value which then transferred to worker id by RX [4]. Packets were distributed to workers evenly, then counted by workers and then dropped. The RX thread was set affinity to core 0. And workers were set to core 2 and core 4 each. All the three cores were on NUMA node 0. The connections of threads is shown as Fig. 4 with the work flow of RX.

Workers were not bottleneck of performance but RX thread was. For light workload, i.e. counting and dropping, for per packet was done by workers, which made the model's RX thread spare no effort to run on its highest rate because rings to workers would never full. So the ability and characteristics of RX thread, i.e. the receiver and balancer, on NUMA could be studied. Section 4 was configured packets of 64, 128, 256, 512, 768, 1024, 1280 and 1500 for testing as section 3. VTune was also used for monitoring the usage of process consuming CPU time.

B. Test Result

The result is shown in Fig. 5. It shows some characteristics as follows.

1) PF_RING ZC VS. DPDK

PF_RING ZC's processing rate appears higher than DPDK's in the pipeline model. For packets of 64, the processing rates, i.e. PF_RING ZC's 9.55Mpps and DPDK's 7.87Mpps, did not reach the line rate. For 128, PF_RING ZC reached line rate 10Gbps, higher than DPDK's 9.73Gbps.

2) Run-to-completion model VS. pipeline model

The pipeline model of PF_RING ZC and DPDK appears run in lower rate than run-to-completion model studied in section 3. The common work of RX threads of pipeline model and run-to-completion model is receiving. The different work of Rx thread of pipeline model is enqueuing packets, exactly packets' pointer as handle, into rings, rather than run-to-completion model's simple statistic and dropping packets.

C. Further Insight

For more microscopic insight into the difference between run-to-completion model and pipeline model. VTune was made use of to measure two model's CPU time consumption. It was recorded that, for packets of 64, the run-to-completion model based on DPDK consumed 78.2% CPU time for polling NIC, 19.9% for freeing buffer memory back to mempool. The ratio of CPU time consumed by pipeline model was 83.5% for polling NIC and 11.4% for enqueuing, without any freeing operations. In spite of DPDK's promotion on the ratio for polling NIC, its pipeline model's processing rate declined to 5.54Gbps, very different from its run-to-completion model's 8.18Gbps. At the same time, ρ for polling NIC increased from run-to-completion model's 9.56%/Gbps to pipeline model's 15.07%/Gbps, **which indicates that the pipeline model's efficiency for polling NIC is lower than run-to-completion model's.**

As to PF_RING ZC's measurement, VTune recorded that it spent 88% CPU time for polling NIC in run-to-completion model but in pipeline model this value came into 43%. There isn't clear insight into closed PF_RING ZC's difference between two models, but a discovery showed by VTune is that **the pipeline model of PF_RING ZC called hash function in a heavier time-consuming way than DPDK.**

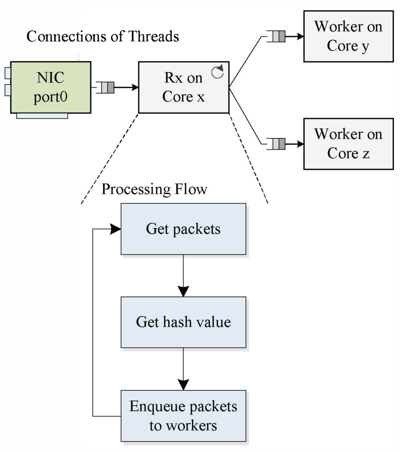


Figure 4. Connections of Threads with Processing Flow of Rx Thread

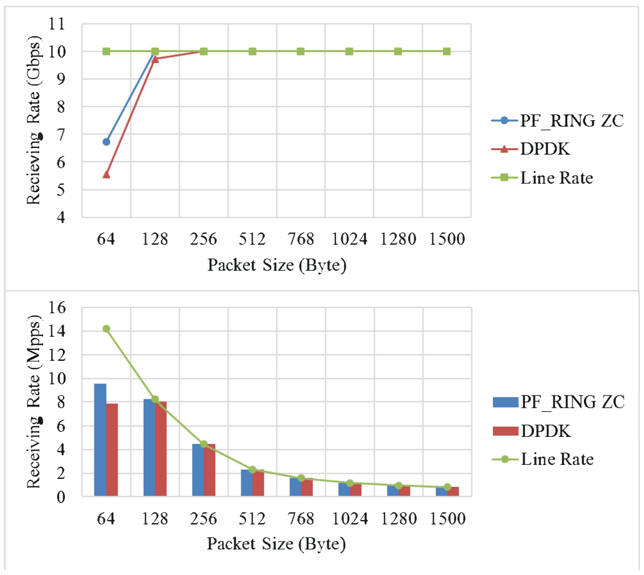


Figure 5. Receiving Rate of Pipeline Model

V. PERFORMANCE OF PIPELINE MODEL ON NUMA

A. Test Description

Relative studies on NUMA shows that the performance of multi-core applications on NUMA varies according to different threads' distribution among NUMA nodes. This section studied the characteristics on how performance of PF_RING ZC and DPDK in pipeline model on NUMA varies with threads' distribution among NUMA nodes.

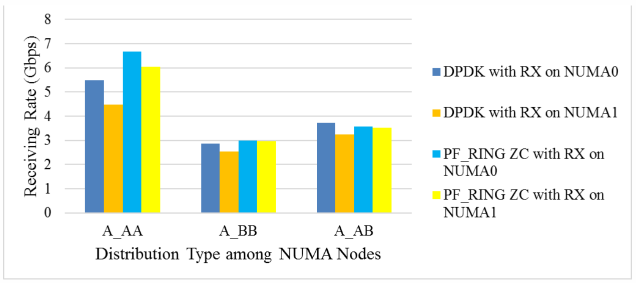
The studied pipeline model in this section is the same as section 4, but threads' affinities with cores varied. The distribution of RX and Workers among NUMA nodes were inducted into three types, marked as *A_AA*, *A_BB* and *A_AB*. A represents node 0 or node 1 of NUMA, so is B. A and B in the same type mark represent different nodes of NUMA. For example, the type mark *A_AA* means that Rx thread and workers are all on node 0 or all on node 1. So 12 testing for PF_RING ZC and DPDK with different distributions were inducted into three distribution type. Every type has two cases, of which RX's affinity set to different nodes each. VTune was made use of to measure the usage of CPU time.

B. Test Result and Analyzition

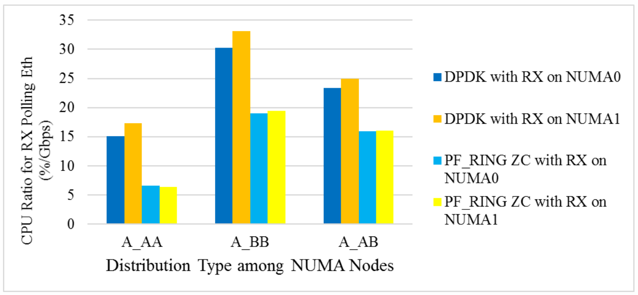
The results is as Fig. 6, which shows receiving rates of 12 testing. Fig. 6 (b) shows ρ for polling NIC and (c) shows ρ for Rx sending data to workers. As referred in section 1, ρ gives the ratio of CPU time consumed on processing per unit packets to evaluate the processing ability, of which the unit is ratio per Gbps, i.e. %/Gbps. Fig. 6 shows the characteristics as follows:

1) Best Distribution Type

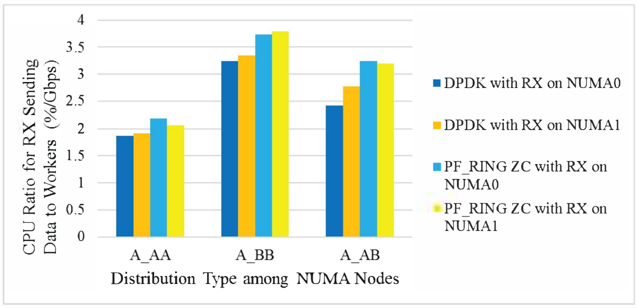
RX works in highest rate when RX and workers are all on the same node of NUMA, of which the distribution type is *A_AA*. The second one is *A_AB*, followed by *A_BB* last. For the difference of applications of different studies, this conclusion looks different with [2]. In fact, this paper coincide with [2] for the impact of operations on remote memory and contention on Last Level Cache (LLC) [11]. For mempool is located on the node of RX thread, and worker threads operate the memory in mempool at the same time, the processing rate of RX decreases with the growth of the number of works on remote node.



(a) Receiving Rate



(b) CPU Ratio for Rx Polling Eth



(c) CPU Ratio for RX Sending Data to Workers

Figure 6. Test Result of Pipeline Models for Different Distribution Type

2) *Best NUMA Node*

It is interesting that RX on node 0 works faster than that on node 1, as to any two cases of same distribution type. There isn't a definite explanation for this conclusion. But it is confirmed that this conclusion isn't contributed by the distribution of threads' stack space among NUMA nodes, which is supported after investigating the difference in processing rates of RX when location of threads' stack space varies on different NUMA nodes. What's more, Fig. 6 shows that DPDK is more sensitive to this conclusion than PF_RING ZC, especially for the ability of polling NIC which is shown by Fig. 6 (b).

3) *PF_RING ZC VS DPDK*

Fig. 6 (a) shows that the application, i.e. a kind of balancer, based on PF_RING ZC in the pipeline is faster than that based on DPDK. Especially for polling NIC, PF_RING ZC works in much better efficient than DPDK, which is shown by Fig. 6 (b). However, DPDK's operations on rings for inter-thread communications is more efficient than PF_RING ZC, shown as Fig. 6 (c). Furthermore, it was found that the API for traffic balancing provided by PF_RING ZC called hash function in a heavy time-consuming way.

VI. CONCLUSION AND OUTLOOK

In the paper, firstly, run-to-completion models in single thread and pipeline models in multi thread were compared on the whole between PF_RING and DPDK separately, after which the difference between PF_RING and DPDK's in these models were concluded with analysis. Furthermore, the characteristics about how performance of PF_RING ZC and DPDK in pipeline model specifically on NUMA varies with distribution type of threads among NUMA nodes were studied, concluded, and made more insight into. For studying above, the new indicator, ρ, giving the ratio of CPU time consumed on processing per unit packets, was put forward and played an important role in above analysis. The paper shows that two frameworks' threads' distributions types among NUMA nodes in the pipeline model, i.e. a kind of load balancing model, contribute to different performance of traffic processing.

As NUMA plays more and more important roles in traffic processing field, the characteristics of multi-core application on NUMA will still be a hot spot. A future work could be conducted for more insights into the pipeline's characteristics and optimization. E.g. the distribution types of threads among NUMA nodes could be extended by setting more RX threads, and the impact between receiving and transmitting in one application could be studied also.

[1] Sebastian Gallenmuller. Comparison of Memory Mapping Techniques for High-Speed Packet Processing[D]. Technical University of Munich, 2014.

[2] Liang ZHU, Hai JIN, and Xiaofei LIAO. SymS: a symmetrical scheduler to improve multi-threaded program performance on NUMA systems[J]. Concurrency Computation: Practice and Experience, 2015, 27(18): 5810-5825. DOI: 10.1002/cpe.3638.

[3] TANG Lu, YAN JinLi, SUN ZhiGang, LI Tao, ZHANG MinXuan. Towards high-performance packet processing on commodity multi-cores: current issues and future directions[J]. Science China Information Sciences, 2015, 58(12): 1-16. DOI: 10.1007/s11432-015-5484-6.

[4] Intel. Intel® 82599 10 GbE Controller Datasheet, Version 3.3, March 2016.

[5] Intel DPDK: Data Plane Development Kit. http://www.dpdk.org/. Last visited 2016-06-17.

[6] Intel DPDK. DPDK User Guides. http://www.dpdk.org/doc/guides/. Last visited 2016-06-17.

[7] Intel VTune. http://software.intel.com/en-us/intel-vtune-amplifier-xe. Last visited 2016-06-17.

[8] PF_RING ZC. http://www.ntop.org/products/pf ring/pf ring-zc-zero-copy/. Last visited 2016-06-17.

[9] ntop. PF RING User Guide, Version 6.1.1, 2015.

[10] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer and G. Carle, "Comparison of frameworks for high-performance packet IO," Architectures for Networking and Communications Systems (ANCS), 2015 ACM/IEEE Symposium on, Oakland, CA, 2015, pp. 29-38. DOI: 10.1109/ANCS.2015.7110118.

[11] Zoltan Majo and Thomas R. Gross. 2011. Memory management in NUMA multicore systems: trapped between cache contention and interconnect overhead. SIGPLAN Not. 46, 11 (June 2011), 11-20. DOI: http://dx.doi.org/10.1145/2076022.1993481